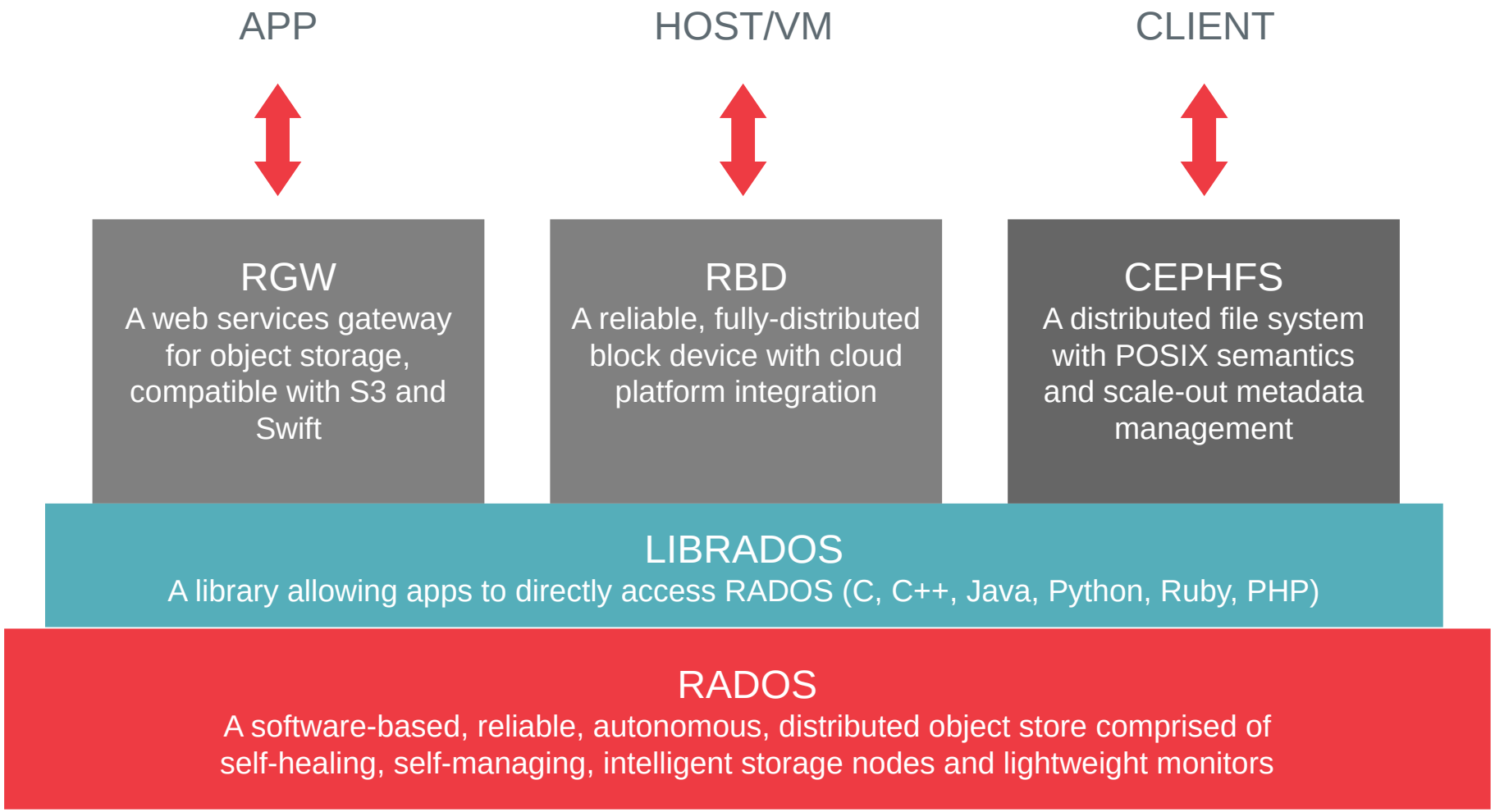
CephFS: Today and Tomorrow

Greg Farnum
gfarnum@redhat.com

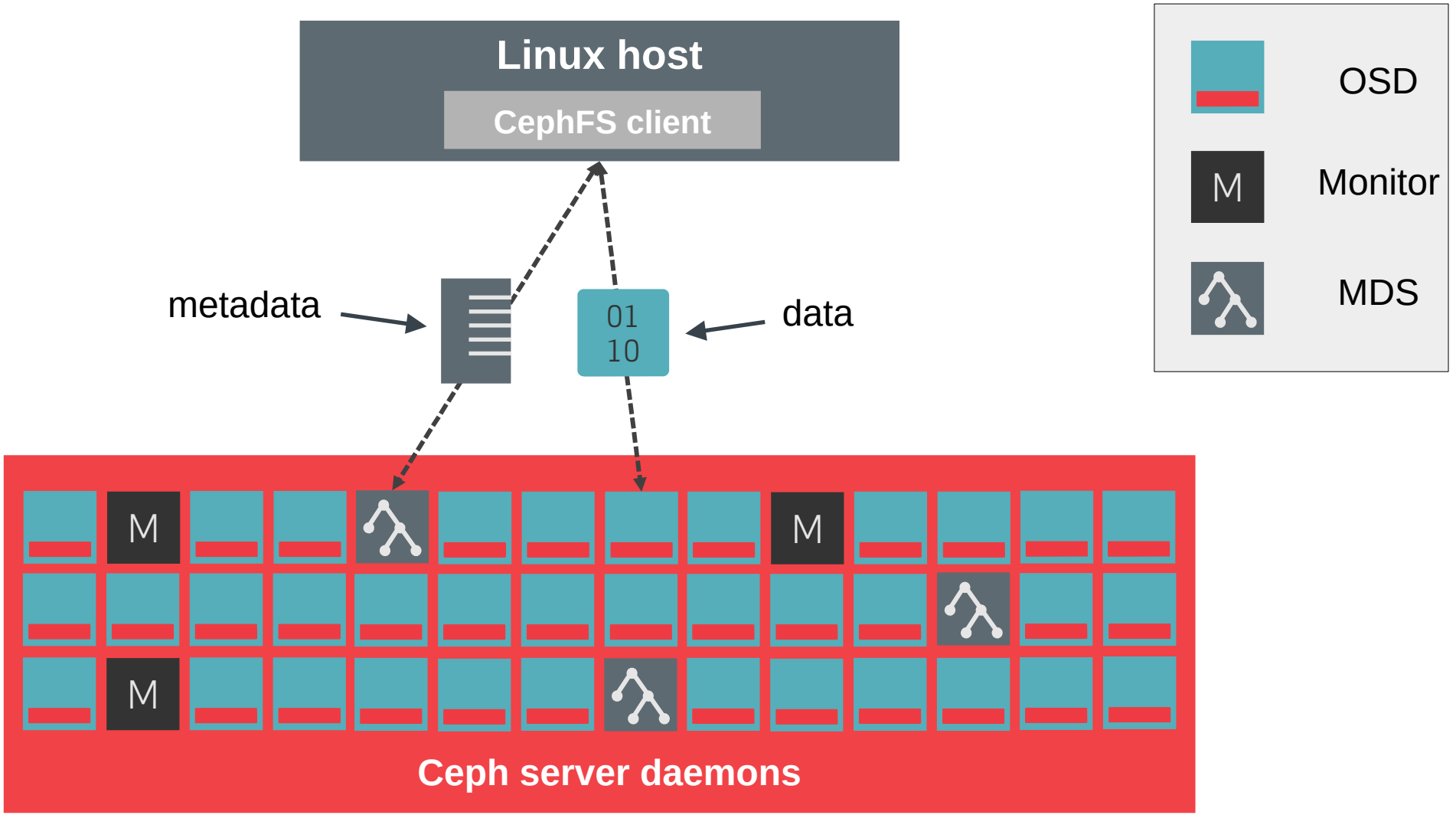
SC '15

Architectural overview

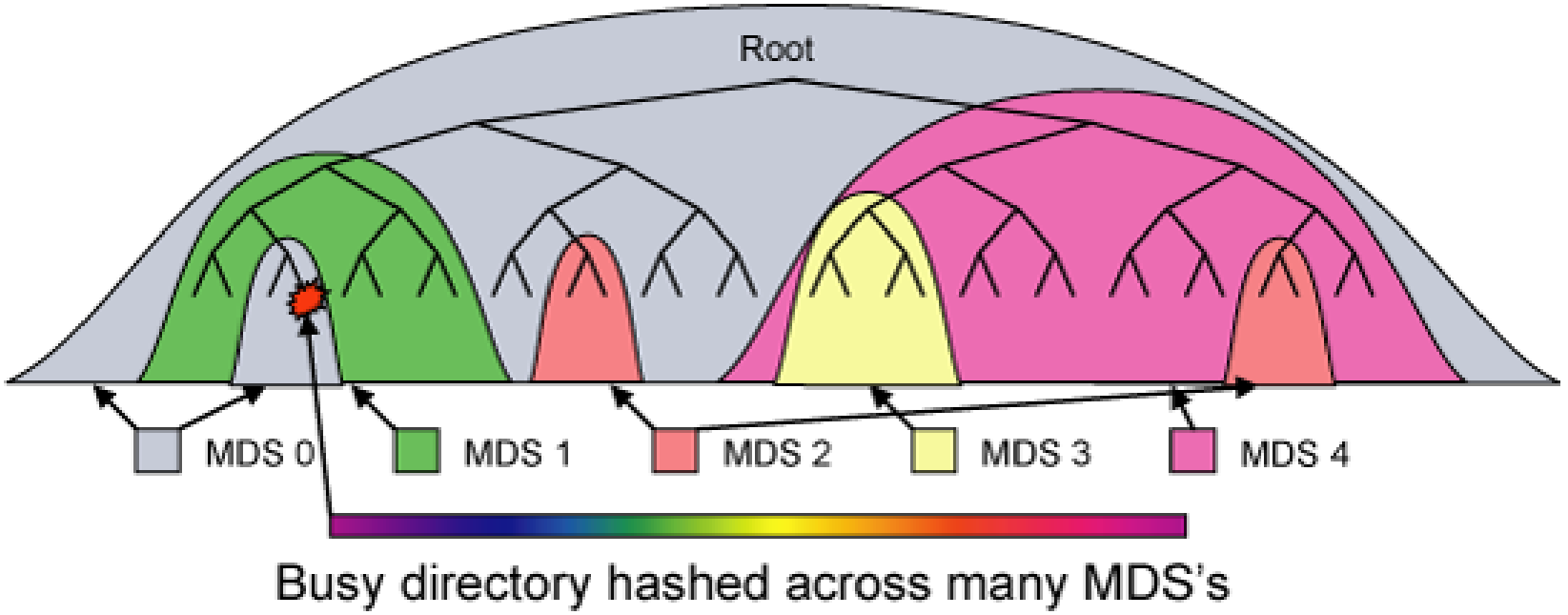
Ceph architecture



Components


Linux host
CephFS client
metadata
01
10
data
OSD
M
Monitor
MDS
M
M
M
Ceph server daemons

Dynamic subtree placement



Using CephFS Today

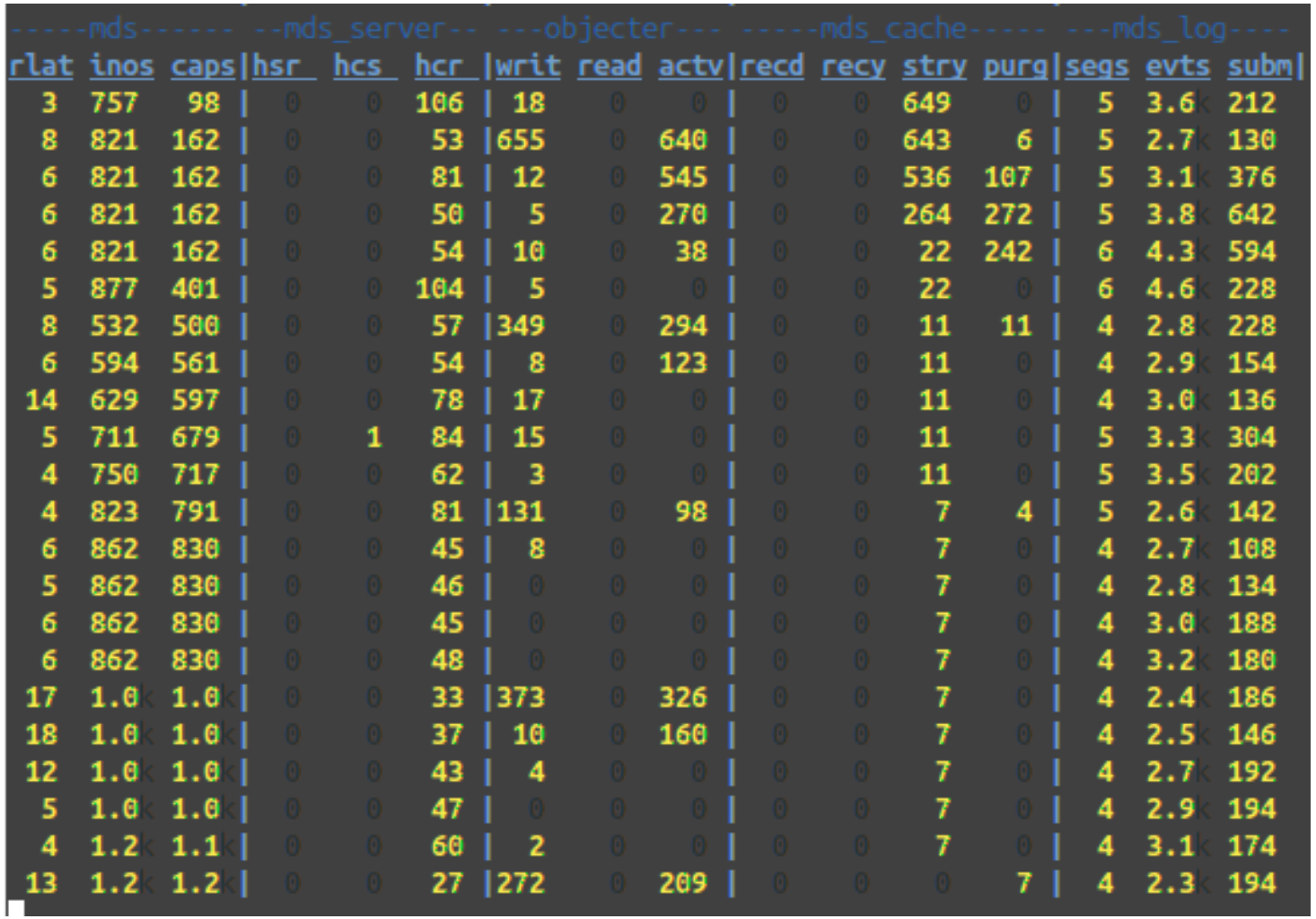
rstats are cool

```
# ext4 reports dirs as 4K
ls -lhd /ext4/data
drwxrwxr-x. 2 john john 4.0K Jun 25
14:58 /home/john/data


# cephfs reports dir size from contents
$ ls -lhd /cephfs/mydata
drwxrwxr-x. 1 john john 16M Jun 25 14:57
./mydata
```

Monitoring the MDS

$ ceph daemonperf mds.a



-----mds------			--mds_server--			---objecter---			-----mds_cache-----				---mds_log----		
rlat	inos	caps	hsr	hcs	hcr	writ	read	actv	recd	recy	stry	purg	segs	evts	subm
3	757	98	0	0	106	18	0	0	0	0	649	0	5	3.6k	212
8	821	162	0	0	53	655	0	640	0	0	643	6	5	2.7k	130
6	821	162	0	0	81	12	0	545	0	0	536	107	5	3.1k	376
6	821	162	0	0	50	5	0	270	0	0	264	272	5	3.8k	642
6	821	162	0	0	54	10	0	38	0	0	22	242	6	4.3k	594
5	877	401	0	0	104	5	0	0	0	0	22	0	6	4.6k	228
8	532	500	0	0	57	349	0	294	0	0	11	11	4	2.8k	228
6	594	561	0	0	54	8	0	123	0	0	11	0	4	2.9k	154
14	629	597	0	0	78	17	0	0	0	0	11	0	4	3.0k	136
5	711	679	0	1	84	15	0	0	0	0	11	0	5	3.3k	304
4	750	717	0	0	62	3	0	0	0	0	11	0	5	3.5k	202
4	823	791	0	0	81	131	0	98	0	0	7	4	5	2.6k	142
6	862	830	0	0	45	8	0	0	0	0	7	0	4	2.7k	108
5	862	830	0	0	46	0	0	0	0	0	7	0	4	2.8k	134
6	862	830	0	0	45	0	0	0	0	0	7	0	4	3.0k	188
6	862	830	0	0	48	0	0	0	0	0	7	0	4	3.2k	180
17	1.0k	1.0k	0	0	33	373	0	326	0	0	7	0	4	2.4k	186
18	1.0k	1.0k	0	0	37	10	0	160	0	0	7	0	4	2.5k	146
12	1.0k	1.0k	0	0	43	4	0	0	0	0	7	0	4	2.7k	192
5	1.0k	1.0k	0	0	47	0	0	0	0	0	7	0	4	2.9k	194
4	1.2k	1.1k	0	0	60	2	0	0	0	0	7	0	4	3.1k	174
13	1.2k	1.2k	0	0	27	272	0	209	0	0	0	7	4	2.3k	194

MDS admin socket commands

- `session ls`: list client sessions
- `session evict`: forcibly tear down client session
- `scrub_path`: invoke scrub on particular tree
- `flush_path`: flush a tree from journal to backing store
- `flush journal`: flush everything from the journal
- `force_readonly`: put MDS into readonly mode
- `osdmap barrier`: block caps until this OSD map

MDS health checks

- Detected on MDS, reported via mon
  - Client failing to respond to cache pressure
  - Client failing to release caps
  - Journal trim held up
  - ...more in future
- Mainly providing faster resolution of client-related issues that can otherwise stall metadata progress
- Aggregate alerts for many clients
- Future: aggregate alerts for one client across many MDSs

OpTracker in MDS

- Provide visibility of ongoing requests, as OSD does

```
ceph daemon mds.a dump_ops_in_flight
{
    "ops": [
        {
            "description": "client_request(client.
            "initiated_at": "2015-03-10 22:26:17.4
            "age": 0.052026,
            "duration": 0.001098,
            "type_data": [
                "submit entry: journal_and_reply",
                "client.4119:21120",
...
```

cephfs-journal-tool

- Disaster recovery for damaged journals:
  - inspect/import/export/reset
  - header get/set
  - event recover_dentries
- Works in parallel with new journal format, to make a journal glitch non-fatal (able to skip damaged regions)
- Allows rebuild of metadata that exists in journal but is lost on disk
- Companion **cephfs-table-tool** exists for resetting session/inode/snap tables as needed afterwards.

Full space handling

- Previously: a full (95%) RADOS cluster stalled clients writing, but allowed MDS (metadata) writes:
  - Lots of metadata writes could continue to 100% fill cluster
  - Deletions could deadlock if clients had dirty data flushes that stalled on deleting files
- Now: generate ENOSPC errors in the client, propagate into fclose/fsync as necessary. Filter ops on MDS to allow deletions but not other modifications.
- Bonus: I/O errors seen by client also propagated to fclose/fsync where previously weren't.

Client management

- Client metadata
  - Reported at startup to MDS
  - Human or machine readable
- Stricter client eviction
  - For misbehaving, not just dead clients
  - Use OSD blacklisting

Client management: metadata

```
# ceph daemon mds.a session ls
...
  "client_metadata": {
    "ceph_sha1": "a19f92cf...",
    "ceph_version": "ceph version 0.93...",
    "entity_id": "admin",
    "hostname": "claystone",
    "mount_point": "\/home\/john\/mnt"
  }
```

- Metadata used to refer to clients by hostname in health messages
- Future: extend to environment specific identifiers like HPC jobs, VMs, containers...

Client management: strict eviction

ceph osd blacklist add <client addr>

ceph daemon mds.<id> session evict

ceph daemon mds.<id> osdmap barrier

- Blacklisting clients from OSDs may be overkill in some cases if we know they are already really dead, or they held no dangerous caps.
- This is fiddly when multiple MDSs in use: should wrap into a single global evict operation in future

FUSE client improvements

- Various fixes to cache trimming
- FUSE issues since linux 3.18: lack of explicit means to dirty cached dentries en masse (we need a better way than remounting!)
- `flock` is now implemented (require fuse >= 2.9 because of interruptible operations)
- Soft client-side quotas (stricter quota enforcement needs more infrastructure)

Using CephFS Tomorrow

Access control improvements (Merged)

- GSoC and Outreachy students
  - NFS-esque root_squash
  - Limit access by path prefix
- Combine path-limited access control with subtree mounts, and you have a good fit for container volumes.

Backward scrub & recovery (ongoing)

- New tool: cephfs-data-scan (basics exist)
  - Extract files from a CephFS data pool, and either hook them back into a damaged metadata pool (repair) or dump them out to a local filesystem.
  - Best-effort approach, fault tolerant
  - In unlikely event of loss of CephFS availability, you can still extract essential data.
  - Execute many workers in parallel for scanning large pools

Forward Scrub (partly exists)

- Continuously scrub through metadata tree and validate
  - forward and backward links (dirs → files, file “backtraces”)
  - files exist, are right size
  - rstats match reality

Jewel “stable CephFS”

- The Ceph community is declaring CephFS stable in Jewel
- That's limited:
  - No snapshots
  - Single active MDS
  - We have no idea what workloads it will do well under
- But we will have working recovery tools!

Test & QA

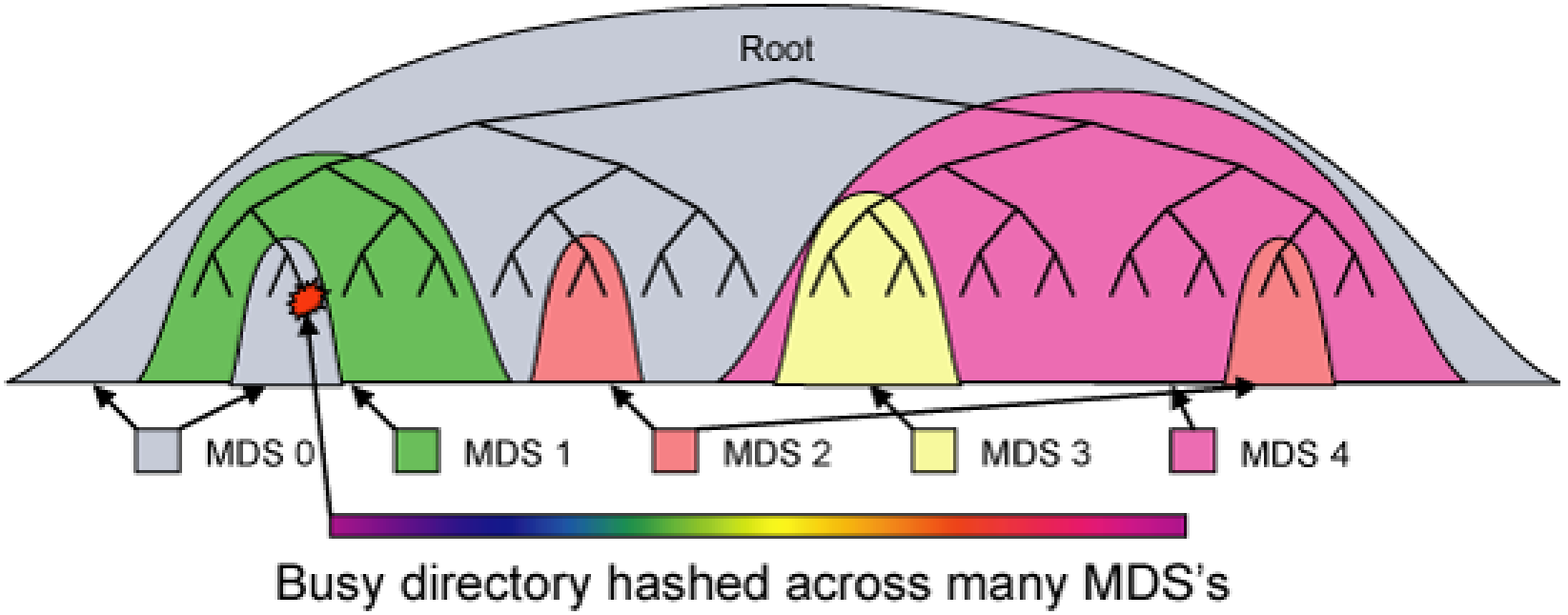
- *teuthology* test framework:
  - Long running/thrashing test
  - Third party FS correctness tests
  - Python functional tests
- We dogfood CephFS within the Ceph team
  - Various kclient fixes discovered
  - Motivation for new health monitoring metrics
- **Third party testing is extremely valuable**

CephFS Future

Snapshots in practice

```
[john@schist backups]$ touch history
[john@schist backups]$ cd .snap
[john@schist .snap]$ mkdir snap1
[john@schist .snap]$ cd ..
[john@schist backups]$ rm -f history
[john@schist backups]$ ls
[john@schist backups]$ ls .snap/snap1
history
# Deleted file still there in the snapshot!
```

Dynamic subtree placement



Functional testing

- Historic tests are “black box” client workloads: no validation of internal state.
- More invasive tests for exact behaviour, e.g.:
  - Were RADOS objects really deleted after a *rm*?
  - Does MDS wait for client reconnect after restart?
  - Is a hardlinked inode relocated after an unlink?
  - Are stats properly auto-repaired on errors?
  - Rebuilding FS offline after disaster scenarios
- Fairly easy to write using the classes provided:

  ceph-qa-suite/tasks/cephfs

Tips for early adopters

http://ceph.com/resources/mailing-list-irc/

http://tracker.ceph.com/projects/ceph/issues

http://ceph.com/docs/master/rados/troubleshooting/log-and-debug/

- Does the most recent development release or kernel fix your issue?
- What is your configuration? MDS config, Ceph version, client version, kclient or fuse
- What is your workload?
- Can you reproduce with debug logging enabled?

Questions?